A New Method of Model Description : Use of Knowledge Base and Inference (データ・セマンティクスの理論と実際に関する研究)

AUTHOR(S):

OHSUGA, SETSUO

数理解析研究所講究録
461 巻 1982 年 408-425

A New Method of Model Description
— Use of Knowledge Base and Inference

by Setsuo Ohsuga

Institute of Interdisciplinary Research,
Faculty of Engineering,
The University of Tokyo

Abstract

A new method for describing and manipulating the engineering models is presented.

Three functions are considered fundamental in model building: model structure definition, model structure modification and model evaluation. The first two functions concern to the structure of the object the user wants to produce while the last function concern to its attributes.

The objective of the paper is to propose a method of model building, by means of which both descriptions on the structure and on the attributes are made in the same framework. Knowledge, represented in terms of the expanded predicate logic, is used for the purpose in combination with the relational database.

1. Introduction

Problem solving is a process to generate and evaluate and often to modify the model of the problem one wants to solve through interactions with computer.[29][30][39] The process lasts until a solution is reached, which satisfies the given conditions.

In this paper, the author discusses a new framework for interactive problem solving systems. In particular, we discuss a new method for building and manipulating the engineering models. We consider that there are three basic functions involved in the model building : model structure definition, model structure modification and model evaluation. The first two functions concern to the techniques to define and manipulate the structure of the object, that is, to define, to represent and to manipulate the constituents of the object and their relations, while the last one has concern to the attributes of the object. Thus, structure and attributes are the major topics we discuss in this paper.

Before going into detail, we will consider two substantial differences in natures between the real object and its model in information.

1.1 Declarative form of representation

Every object in the real world shows its own attributes. An object as the physical existence and its attributes are inseparable. Suppose, for example, some one is making a box, given a set of boards. As soon as he makes the box, the attribute 'volume' occurs, which has not been existing before. On the other hand, no model in information today has this nature but man has to give the description on attributes every time he defines a new object. In other words the object structure and the attributes are independent to each other in the model world. Surely, this is the defect of the model. The author believes that this is not the intrinsic nature of the model itself but because of our software technique for model building being not fully developed, yet. We need therefore to develop a new method to improve the situation.

The further investigation on this problem leads us to the conclusion that both model structure and attribute have to be represented in the declarative form. The reason is as follows:

In most cases, today, the model structure is represented in the data structure while some attributes are given in the form of procedure to evaluate their values. This style of representation involves, at least, two defects in establishing automatically the object-attribute relationship. First, the attributes referred to in a process are only part of all what the object possesses. They are selected in relation to the design goal. Hence, the evaluation procedure tends to be task dependent, while the object-attribute relationship is essentially task independent in nature. Second, it is difficult to design a system that generates attributes in the form of procedure automatically every time a new object is defined.

Hence the attribute also has to be represented declaratively. This is the major topics discussed in this paper.

1.2 Component structure and complete structure

Second, we note that, in the model world, we can define and manipulate any conceptual object as we like (under the assumption that we have a good tool for building models), while we can not do for some kinds of real objects. Consider, for example, one wants to design a new chemical compound. In information, we can define any model and can access directly to any constituent in it, while we can use only a limitted number of methods, in the real world, to treate the compounds, such as: heating, adding catalyzer, radiating and so on. Some bondings between atoms in the compound will change by these processes but some others may remain unchanged. Then, the content ratios of new and old compounds and their distribution in the object may affect the properties of the products.

This notion drives us to further requirements to the real model building system. First, it has to possess the facility to simulate the effect of the external means of processing object to the change in its component structure. Second, at least two level structure is necessary for modelling the object composed of the mixture of different kinds of constituents: micro structure to represent the components and macro structure to represent the complete object. In general, the former has a rather simple, deterministic structure while the latter has a complex and, sometimes, probabilistic structure. In fact, each structure may require more than two levels. In this paper, however, we concentrate our effort to the issue presented in the section 1.1 and we do not touch on above problem because that one is more fundamental than this one.

The primary object of this paper is, then, to have a system that integrates three functions of model building into the same framework

so that the user can use these functions in any order he likes.

2. Knowledge Based System Framework

2.1 Knowledge based system as the core of engineering design system

In many real problems, the data structure has been used to represent the object structure. There is, however, no solid theoretical foundation nor unified principle on what data structure to use to the problem. Hence the user has to decide himself what data structure ad hoc and has to write down programs to process it.

Alternatively some people intend to use the conventional database for engineering purpose. Then he can use a set of functions the database system provides him. Its defect, however, is that the structure today's data model can support is so simple that he often feels inconvenience in representing the object with complex structure. In both cases, the model structure is unrelated to its attributes.

Taking this situation and our requirements for the model building presented before into account, we define,in the following, a formalism for representing attributes declaratively, together with the structure of objects. We need, then, a new sophisticated method of representation and we propose a knowledge base system for the purpose. In short, we use an extended version of the predicate logic as the main framework to represent the attributes of object together with its structure. In the following sections, we outline the framework of this system and show how the system works.

2.2 Organization of knowledge based interactive system

Fig.1 shows the organization of the knowledge based interactive

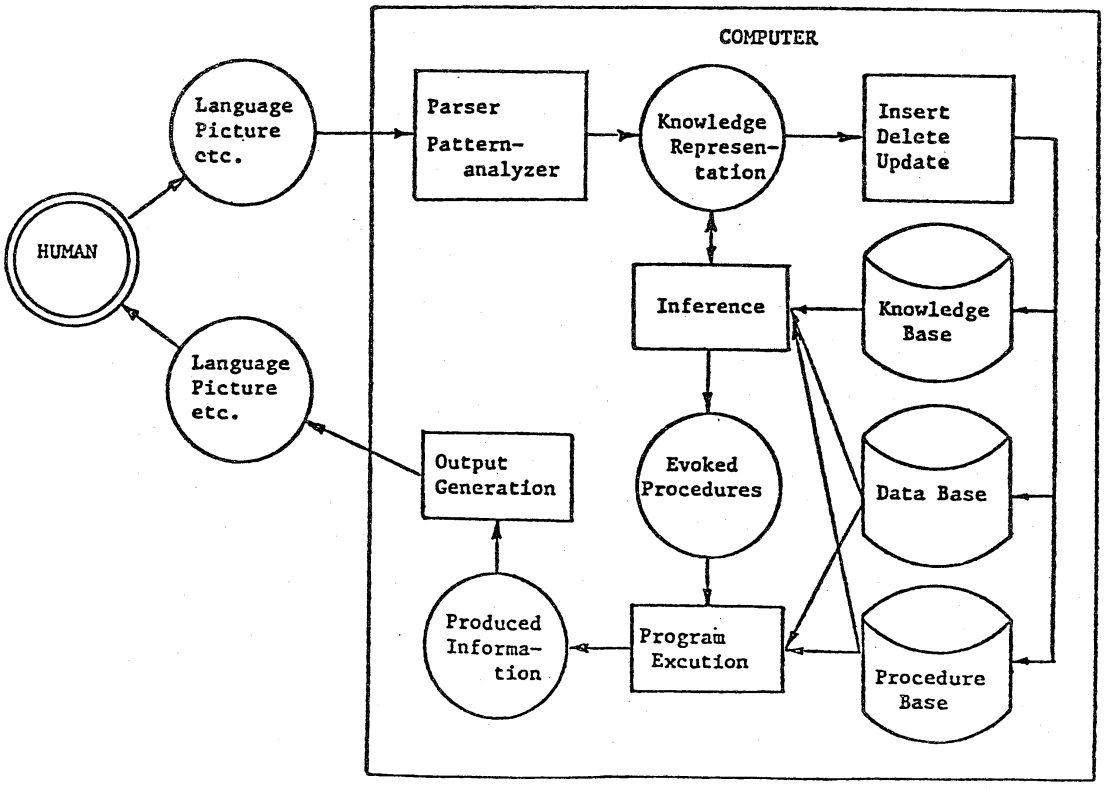


Fig.1 Knowledge based interactive system organization

system we are working to realize, named KAUS (Knowledge Acquisition and Utilization System).[32][33] The system is designed to adapt to wide class of applications.

There are three different information bases: the knowledge base, the data base and the procedure base. The knowledge base (KB) includes a collection of task specific knowledge as well as general rules, theories, facts etc.. The knowledge is represented in a special form called the knowledge representation. The database(DB) is the collection of data. The relational database is considered here because of its basic simplicity and generality.[7][43] The procedure base (PB) is a collection of built-in procedures. The procedures are evoked at the evaluation stage of the request by the inference mechanism.

In the figure, a circle denotes information in the non-procedural form while a block denotes the procedure that converts the information from one form to another. The user uses language and/or picture to express his request. It is transformed into the intermediate language (knowledge representation) by the parser. Then the inference mechanism transforms the request into another form until it can be resolved into a set of built-in procedures, which, in turn, generate answers. The result is displayed by the output generator.

3 Representation of Knowledge and Inference

3.1 Predicate logic

Various knowledge representation languages or schemes have been proposed including the production rule[28][41], the semantic network [17][27], the conceptual dependency[40], KRL[23], the predicate logic [14][19] and so on. If we ignore the details and the difference in the implementions, however, the objects and some of the basic functions are common to all approaches. The difference, if any, is in the logical power, that is, one can reresent and manipulate a concept while the other can not. The logical power depends on what basic concepts they involve. In the following, we will show how it depends on the basic concepts by a few examples. We use the predicate logic as the representation of knowledge for this purpose.

We begin with an example expression:

"If the melting point of a metal is lower than a temperature, then it is in the liquid state at the temperature." ($1)

The meaning of this expression is the complex of the primitive meanings of the simple sentences composing the complex sentence. There is close relation between the syntax rule to compose a complex sentence from the simple sentences and the semantics to define the meaning of the complex sentence from those of the simple sentences. This is formalized by means of the predicate logic.

Let the expression ($2) above is denoted S. Then it is decomposed, by introducing variables properly.

We define the following predicates;

(MELT-POINT x y) : melting point of x is y ($2)
(LESS-THAN y z) : y is less than z ($3)
(LIQUID x z) : x is in the liquid state at the temperature z ($4)

By substituting a constant into a variable in these predicates, we obtain a proposition. For example,

(MELT-POINT #Cu #1084.5) : melting point of #Cu is 1084.5(°C) ($5)
(LESS-THAN #1084.5 #1100) : 1084.5 is less than 1100 ($6)

are both true propositions. The symbol # denotes that the term is constant.

In the system, the so called many sorted logic[26] is adopted as the basis, which is, then, expanded in the latter section. In this logic, the predicate is defined to represent 'the relation between objects in respective domains'. The collection (or the set) U, of all objects which we have interests is called the universe. Then a variable is defined over the certain subset of U, which we call the domain. Then we formalize the expression S,

$$(\forall x/\text{metal})(\forall y/\text{real})(\forall z/\text{real})[(\text{MELT-POINT } x\ y)\cap(\text{LESS-THAN } y\ z) \Rightarrow (\text{LIQUID } x\ z)] \quad (\$7)$$

where $(\forall x/\text{metal})- - -(\forall z/\text{real})$, called the prefix, means, for all x in the domain 'metal', for all y in the domain 'real' and for all z in the domain 'real'. The symbols $\cap$ and $\Rightarrow$ are used to mean 'and' and 'imply' respectively. Besides then, $\cup$, $\sim$ and $\Leftrightarrow$ are used meaning 'or', 'not' and 'equivalent' respectively. $\forall$ and $\exists$ are the quantifiers meaning 'for all' and 'for some'. The formalized expression such as ($7) is called the well formed formula or, simply, formula and each component (predicate) is an atom. A formula is defined formally as follows[4] ;

(1) An atom is formula
(2) If P and Q are formula's, then $P\cup Q$, $P\cap Q$, $\sim P$, $P\Rightarrow Q$ and $P\Leftrightarrow Q$ are formulas,
(3) If P is formula and x is a variable in P defined over the domain X, then, $(\forall x/X)P$ and $(\exists x/X)P$ are formulas
(4) formulas are generated only by a finite number of applications of (1) through (3).

3.2 Inference

Suppose a query;"Is the cupper in the liquid state at 1100°C?" is presented to the system holding the formulas, ($5), ($6) and ($7). It is represented (LIQUID #Cu #1100(°C))? There is no information that matches directly with the query. However, by substituting #Cu and #1100 into x and z of ($7) respectively (it is possible because ($7) holds for every x and z), the consequence-part of ($7) matches with the query. Therefore, if its condition-part is evaluated and proved true, then the query is also true. That is

$$(\exists y/\text{real})[(\text{MELT-POINT } \#\text{Cu } y)\cap(\text{LESS-THAN } y\ \#1100)] \quad (\$8)$$

is deduced to replace the original query. The quantifier of y changes from $\forall$ to $\exists$ in this process. We can prove that the formula ($8) is true by using the formulas ($5) and ($6) and by substituting 1084.5(°C) into y. This process is the inference. The inference algorithm for many sorted logic is generalized as shown in Appendix A1.

3.3 Database access and program execution

The following points are to be considered at the design of the inference mechanism in the real knowledge based system.

First, in the example of ($7), we assumed that the very simple relations between individual objects such as ($5) are given as the formulas. It is, however, impractical. There can be a good number of similar relations for possible combinations of objects. Usually they are collected and represented in the different form, e.g., in the tabular form. This is very the relational database. For example, the collection of the formulas such as (MELT-POINT #Cu #1084.5), is replaced by the relation ALPHA(metal, melting point) shown in Fig.2 causing the separation of a portion of the knowledge

Ag	961.9
Au	1064.4
Cr	1890.0
Fe	1535.0
Mn	1244.0
.	.
:	:
Ni	1455.0

Fig.2 A relation

from the main body. Then, provision has to be made to link the knowledge base and the database at the run time as shown in Fig.1. Providing the system with such a capability of dynamic linking will enable the user to use the commercial databases through the knowledge base system.

Second, the formula (LESS-THAN #1084.5 #1100) in ($6) is still impractical to represent not only in logical form but even in the form of database, for there are infinite possible combinations of the real values. We begin with the second problem.

The predicate ($6) was necessary to evaluate the corresponding predicate in the formula ($8) through the matching function of the inference mechanism. But, because (LESS-THAN a b) is the well defined relation, we can do it more efficiently by providing the system with the specific procedure to do it. It also abbreviates the necessity for providing formulas ($6). Then provision also has to be made to hold the specified procedure and evoke it in the evaluation process at the run time.

There are a number of similar predicates to the one above other than binary relations; arithmetic functions such as x+y=z denoted (ADD x y z), mathematical functions such as sinx=z denoted (SIN x z) and so on. We prepare a procedure to evaluate each of the predicate. The collection of the procedures forms the program base of Fig.1. The procedure is evoked by the inference mechanism when the predicate appears in the query and becomes evaluable. A predicate is evaluable if independent variable(s), specified in advance to every predicate (underlined in above examples),is substituted by the specific value(s). The variable(s) besides the independent variable(s), if any, is the dependent variable(s). Then the procedure is defined to make computation using the independent variable(s) and then to substitute the result(s) into the dependent variable(s). For example, the procedure defined to the (ADD x y z) evaluates (ADD 2 6 z) to be true and substitute 8 into z. We call these predicates the procedural predicate. They are dealt with as the ordinary predicates until it becomes evaluable, but after then they turn to the procedure calls.

Hence we can solve the second problem by designing the inference mechanism to involve the capabilities to look for the evaluable procedural predicates in the query and to evoke the corresponding procedure.

In the first case, on the other hand, the linking is performed by the normal inference operation by providing a special predicate for each relation(file). Let "relation u contains a tuple (x y,...,z)" be a predicate and be formalized to (RELATION u; x y...z)(Fig.2). Then, the relation and its meaning, "melting point of x is y", are linked to form the formula,

$$(\forall x/\text{metal})(\forall y/\text{real})[(\text{RELATION } \underline{\#\text{ALPHA}} : x\ y) \Rightarrow (\text{MELT-POINT } x\ y)], \quad (\$9)$$

meaning that "if the relation ALPHA contains the tuple (x, y), then it means that the melting point of x is y". The RELATION is the procedural predicate to which the procedure is defined to read the file u, where u is a variable of which the value is the identifier of the relation. The predicate RELATION turns to evaluable when the variable is substituted by the specific identification code. Hence, if 'the melting point of x' is asked, the inference algorithm changes the request into the file access procedure using the formula ($9). In case of the example before, the inference mechanism will deduces

$$(\exists y/\text{real})[(\text{RELATION } \#\text{ALPHA} : \#\text{Cu } y) \cap (\text{LESS-THAN } y\ \#1100)] \quad (\$10)$$

from ($8) and ($9).

In this formula, the procedure for (RELATION #ALPHA :#Cu y) can be executed any time because ALPHA is fixed while (LESS-THAN y

#1100) is deferred executing until y will have been obtained by the former operation. Thus, the inference mechanism also takes charge of determining the order of execution among the procedural predicates. This technique of linking the knowledge base and the database is quite effective as follows;

First, note that the directed graph can be represented in the table form and is manipulated by the logic system via the formula,

$$(\forall x/\text{node})(\forall y/\text{node})[(\text{RELATION \#ARC} : x\ y) \Rightarrow (\text{ARC}\ x\ y)] \qquad (\$11)$$

where the relation #ARC involves all directed arcs of the graph as the ordered pairs of nodes (Fig.3) and the predicate (ARC x y) means "there is a directed arc from the node x to y." This will be referred to again in the later section.

$$(\forall x)(\exists y)[(\text{RELATION \#ARC}: x\ y) \Rightarrow (\text{ARC}\ x\ y)]$$

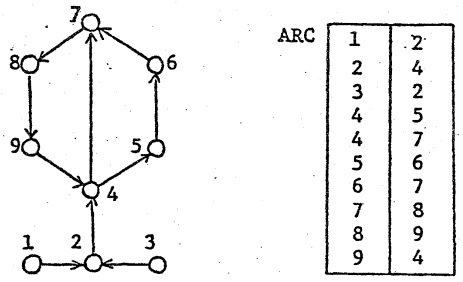


Fig.3 A graph and its representation

Second, though the formula ($9) has been derived as a definition of the static file, this idea is extended easily to the dynamic file. Suppose, given a request, a set of data that satisfy the request is obtained in the temporary file. Let the request be of the form (Q x y ...z). Then, by definition, the description of the temporary file is

$$(\forall x/X)\ldots(\forall z/Z)[(\text{RELATION \#TF} : x\ldots z) \Rightarrow (Q\ x\ldots z)] \qquad (\$12)$$

Through this formula the intermediate results becomes available for latter processing same as the static files. Then, the system allows the user to set arbirary subgoals in his goal directing work. It ensures the results of the preceeding stage being used as the start of the next stage.

Third, assume the situation where a separate relation (file) corresponds to every item of the parent relation, which involves more detailed information on the item, thus forming the structure of files (Fig.4). Suppose also the relation between an item in the parent relation and the identifier of its child relation is given in the other relation, say, BETA, in the form of BETA (item(s) of the parent relation, child relation identifier). For example, assume the parent item be a tuple (x y) and the child relation w composed of the tuples, (u, v), corresponds to x. Let (C-FILE x w) be the predicate meaning "the identifier of the child relation corresponding to x is w". Then by using the formula

$$(\forall w/\text{id})(\forall x/X)[(\text{RELATION \#BETA}:x\ w) \Rightarrow (\text{C-FILE}\ x\ w)] \qquad (\$13)$$

the inference mechanism connects dynamically the parent and children relations. This is not the very special but rather the normal situation that may occur frequently in the design process, that is, in the process to make instances of the structured objects from the materials and to select the one among them that has the given property. In many cases the databases may be used to represent the property of the structures. The object itself has to be represented by a data structure. Fig.4 shows a simplest case where the structured object is represented by an item

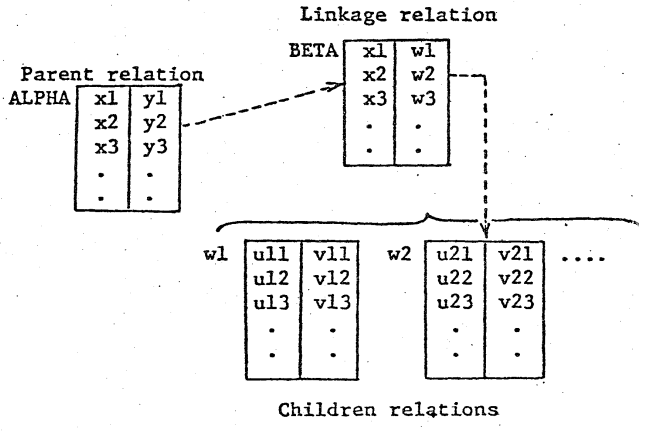


Fig.4 Hierarchical structure of relations

of the parent relation, to which a child relation corresponds to represent the property.

3.4 Conversion to the relational algebra

More than one RELATION predicates often appear in the formula in the various forms. A processing rule is specified to each form as shown in Table 1. Each rule in this table denotes that the formula in the left is transformed to the one in the right.

(i) (RELATION F: x y)∩(RELATION G: x y) = (RELATION F∧G: x y), if $F_{\wedge}G \neq \emptyset$
(ii) (RELATION F: x y)∪(RELATION G: x y) = (RELATION F∨G: x y), if $F^{\vee}G \neq \emptyset$
(iii) (RELATION F: x y)∩(RELATION G: y z) = (RELATION F*G(y): x y z), if F*G(y)≠∅
(iv) ∼(RELATION F: $\overline{x}$)∪(RELATION G: $\overline{x}$) = (RELATION G : $\overline{x}$) , if G⊃F
(v) (∃x/X)(RELATION F: x y) = (RELATION #F(x): y), if #F(x) ≠∅
(vi) (∀x/X)(RELATION F: x y) = (RELATION /F(X): y), if /F(X) ≠∅

where F∧G ;'intersection' of F and G,
F∨G ;'union' of F and G,
F*G(x);'join' of F and G with respect to x,
#F(x) ;'projection of F to delete x,
/F(X) ;'division' of F by X

Table 1 Conversion rule from logic to relational algebra

For example,
(RELATION F : x y)∩(RELATION G : x y)=(RELATION $F_{\wedge}G$: x y)
denotes that the conjunction of two RELATION predicates, both involving the same tuples, is logically equivalent to the single predicate involving the relation obtained as the intersection of original relations. Here the relation is regarded as the set of tuples. Note that each rule gives the logical equivalence, that is, equality in the logical value. The if-part in the right side of each rule shows the condition for the formula being true. The table, therefore, gives rules to transform the logical evaluation process into the set-theoretical or relational algebraic operations.

4 Extention of the Framework

4.1 Multi layer logic

Though the many sorted logic possesses a number of remarkable features as the tool for representing knowledge, its mechanism is still too simple to represent every concept involved in many real applications. We show it referring to an example.

Consider an expression "the average of u is v" and denote it (AVERAGE u v). The syntax seems to be the same as "the melting point of x is y" denoted (MELT-POINT x y). There is, however, a substantial difference in the mathematical structure between them. The meaning of the word 'average' is defined as the operation that maps a set u into a real value v. Hence, u has to be a set in (AVERAGE u v).

Suppose, #A is a subset of U involving only integers, say, #A={2, 4, 6, 8}. Then #A can be substituted into u resulting in (AVERAGE #A, 5), a true proposition, while a single element, say #5, can not be substituted. To the contrary, a simple element, say #Cu, can be substituted into x of (MELT-POINT x y) while a set {#Cu, #Ag,...} can not. In the mathematical term, this is the

difference between $u \subset U$ and $x \in U$. In order to accomodate both of them in the same framework, we need further expansion of the formalism introducing a new mathematical concept, the powerset, 2^U, of U and expanding the universe from U to $U \vee 2^U$, where the symbol $\vee$ means "union" of sets. Then we rewrite $u \subset U$ as $u \in 2^U$ which is of the same structure as $x \in U$ and apply it the many soted logic. For example, the expression "for every (set) u, there is the average of u" can be represented

$$(\forall u/2^U)(\exists v/\text{real})(\text{AVERAGE } u\ v) \qquad (\$14)$$

In fact, the empty set should be excluded from the powerset. In the following, we denote the powerset but the empty set being excluded as *U introducing the new symbol *. This expansion in the syntax induces a side effect. By definition, some variable u is an element of *U in a predicate and, at the same time, can be a domain of the other variable(s). For example, "some element of a set is less than the average of the set" is represented,

$$(\forall u/{*U})(\exists x/u)(\exists y/\text{real})[(\text{AVERAGE } u\ y) \cap (\text{LESS-THAN } x\ y)]. \qquad (\$15)$$

where u is an element of *U and at the same time the domain of x.

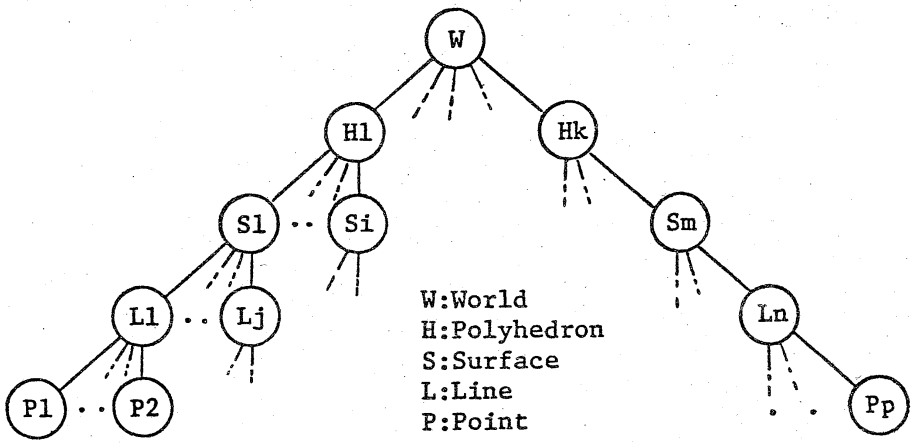


Fig.5 A hierarchical structure

It is easy to extend the idea upward to define a high order powerset denoted, $*^n U$, $n>0$. It is defined recursively, $*^n U = *(*^{n-1} U)$. It implies that the predicate logic is expanded to include the hierarchically structured entity as the objects of description. In the hierarchical structure, a n-th level entity is composed of n-1th level entities as shown in Fig.5. The set of the 0-th level entities are referred to the base set. Any substructure of the structure is itself an entity. If, in Fig.5, level0, level1, level2 and level3 entities represent the point, the line, the surface and the polyhedron respectively, then the structure represents a three dimensional geometric model. Arbitrary description can be made on the structure or any of its substructure. For example, the expression "every polygon of the polyhedron H1 has some line of length a" is represented

$$(\forall x/\ H1)(\exists y/x)(\text{LENGTH } y\ a) \qquad (\$16)$$

The more interesting is the case where the structure is incompletely specified. For example, consider the expression, "Is there a polyhedron composed of the set of vertices V, of which every surface has a line of length a?" This is asking a structure that satisfies the given condition (just the basic pattern of the design process!). Since the polyhedron is an element of * V (as shown below), it is represented as,

$$(\exists u/{*^3 V})(\forall x/u)(\forall z/x)(\exists y/x)[(F_L\, z) \cap (F_S\, x) \cap (F_H\, u) \cap (\text{LENGTH } y\ a)]? \qquad (\$17)$$

*V denotes a set of all subsets of vertices in V. A line is an

element of it, that satisfies the condition as the lines (the number of vertices included in the element is just two). F_L represents the condition. $*^2V$ denotes a set of all subset of $*V$. A surface, if any, is an element of it, that satisfies the condition as the surface (composed of lines, lines in the element form the closed link and are co-planar, and so on). F_S represents it. $*^3V$ denotes a set of all subset of $*^2V$. A polyhedron, if any, is one of its element that satisfies the condition as the polyhedron (composed of surfaces which forms a closed body, and so on) which is represented as F_H. In the formula (\$17), the variables u, x and y represents the polyhedra, the surface and the lines respectively. We call the extended version of the predicate logic multi-layer logic.

4.2 Property of multi layer logic

Multi-layer logic has number of interesting properties. It contains, as has been stated, two ways in representing hierarchical structure of an object; a specific structure as Fig.5 and a variable structure including power-set operator * as in (\$17). A logical formula is evaluated in the different way depending on which type of structure description is included. In the following, these cases are referred to separately as specific structure and non-specific structure respectively.

Consider a formula

$$(Q_x^n x^n/X)(Q_x^n x^{n-1}/x^n) - - - (Q_x^o x^o/x^1)(Q_Y \mathbf{y}/Y)(M\ x^m, \ldots, x^\ell\ \mathbf{y}) \qquad (\$18)$$

where Q_x^r, r=n, ..., 0, is either $\forall$ or $\exists$, X is a hierarchy involving n+1 level, $(Q_Y \mathbf{y}/Y)$ denotes another hierarchy shown in the abbreviated form and $(M\ x, \ldots, x^\ell\ \mathbf{y})$ is the main body, usually called the matrix, of the formula in which the level m of x^m is the highest among the variables, $x^m, \ldots, x^\ell$, and l, the level of x^ℓ, is greater than or equal to zero. By definition, any level is permitted to n as long as n≥m. That is, the levels of variables appearing in the prefix and the matrix can be, in principle, independent to each other. However, the formula is, in fact, reduced, by the following theorem, to the one in which the level of the variables are related to each other. In the following, we consider the hierarchical structure being composed of a finite base set.

Theorem 1 (specific structure); Let the specific structure X of level n+1 be

$$X=\{A_1^n, A_2^n, \ldots, A_S^n\} \qquad (\$19a)$$

and for any s, 0<s≤n

$$A_{i\ldots j}^s = \{A_{i\ldots j_1}^{s-1}, \ldots, A_{i\ldots j_*}^{s-1}\} \qquad (\$19b)$$

$$A_{i\ldots\ell}^{m+1} = \{A_{i\ldots\ell_1}^m, \ldots, A_{i\ldots\ell_p}^m\} \qquad (\$19c)$$

Then

$$(Q_x^n\ x^n/X) - - - (Q_x^m\ x^m/x^{m+1}) - - - (Q_x^o\ x^o/x^1)(Q_y\ \mathbf{y}/Y)(M\ x^m, \ldots, x^\ell, \mathbf{y})$$

$$= \begin{cases} (\forall x^m/C)(Q_x^{m-1}x^{m-1}/x^m) - - - (Q_x^o x^o/x^1)(Q_y \mathbf{y}/Y)(Mx^m, \ldots, x^o, \mathbf{y}) \\ \qquad \text{if } (Q_x^n, \ldots, Q_x^m) = (\forall, \ldots, \forall) \qquad (\$20a) \\ (\exists x^m/C)(Q_x^{m-1}x^{m-1}/x^m) - - - (Q_x^o/x^1)(Q_y \mathbf{y}/Y)(Mx^m, \ldots, x^o, \mathbf{y}) \\ \qquad \text{if } (Q_x^n, \ldots, Q_x^m) = (\exists, \ldots, \exists) \qquad (\$20b) \\ (\forall x^{m+1}/B)(\exists x^m/x^{m+1}) - - - (Q_x^o x^o/x^1)(Q_x \mathbf{y}/Y)(Mx^m, \ldots, x^\ell, \mathbf{y}) \\ \qquad \text{any other cases} \qquad (\$20c) \end{cases}$$

where B and C are the structures of level m+2 and m+1 respectively. B is composed, using $A_{i\ldots}^m\ell$p's by the following procedures;

Let

$$X' = \begin{cases} B_1^n \wedge B_2^n \wedge \ldots \wedge B_S^n & \text{if } Q_x^n = \forall \\ B_1^n \vee B_2^n \vee \ldots \vee B_S^n & \text{if } Q_x^n = \exists \end{cases}$$

$$B_{i\ldots j}^s = \begin{cases} B_{i\ldots j_1}^{s-1} \wedge \ldots \wedge B_{i\ldots j_k}^{s-1} & \text{if } Q_x^{s-1} = \forall \\ B_{i\ldots j_1}^{s-1} \vee \ldots \vee B_{i\ldots j_k}^{s-1} & \text{if } Q_x^{s-1} = \exists \end{cases} \qquad m<s\leq n$$

$$B_{i\ldots\ell}^{m+1} = A_{i\ldots\ell_1}^m \wedge \ldots \wedge A_{i\ldots\ell_p}^m \quad \text{if } Q_x^m = \forall$$

$\backslash A^m_{i\cdots\ell_1} \vee \ldots \vee A^m_{i\cdots\ell_p}$ if $Q^m_x = \exists$

X' is then rearranged into the conjunctive normal form

$$X' = (C^m_{11} \vee \ldots \vee C^m_{1\nu_1}) \wedge \ldots \wedge (C^m_{u1} \vee \ldots C^m_{u\nu_n}) \qquad (\$21)$$

Then B is defined as a set of level m+2, $B=\{B^{m+1}_1, \ldots, B^{m+1}_u\}$, where $B^{m+1}_i = \{C^m_{i1}, C^m_{i2}, \ldots, C^m_{i\nu_i}\}$, i=1, 2, ..., u.

In case of $(Q^n_x, \ldots, Q^m_x)$ being $(\forall, \ldots, \forall)$ or $(\exists, \ldots, \exists)$, X' reduces to $X' = A^m_1 \wedge \ldots \wedge A^m_w$ or $X' = A^m_1 \vee \ldots \vee A^m_w$ respectively, where A^m_i, i=1,...,w denotes an element of the set of all $A^m_{i\cdots\ell_q}$'s. Then,

$C = \{A^m_i \mid i=1,\ldots,w\}$

Theorem 2 (non-specific structure);

$$(Q^n_x x^n / {*^n} Z) - - - (Q^m_x x^m / x^{m+1}) - - - (Q^0_x x^0 / x^1)(Q_y y / Y)(M\ x^m, \ldots, x^\ell\ y)$$
$$= (Q^n_x x^n / {*^m} Z)(Q^{m-1}_x x^{m-1} / x^m) - - - (Q^0_x x^0 / x^1)(Q_y y / Y)(M\ x^m, \ldots, x^\ell\ y) \qquad (\$23)$$

4.3 Inference algorithm for multi layer logic

Theorem 1 and 2 tells us that any formula in multi layer logic, except the one of ($20c) is reduced to the one, of which the level of variable in the prefix is at most the same as that in the matrix. The formula ($20c), in fact, can further be converted to

$$(\exists x^m / B^{m+1}_1) - - - (Q^0_x x^0 / x^1)(Q_x y / Y)(M\ x^m, \ldots, x^\ell, y) \cap - - -$$
$$- - - \cap (\exists x^m / B^{m+1}_u) - - - (Q^0_x x^0 / x^1)(Q_x y / Y)(M\ x^m, \ldots, x^\ell, y) \qquad (\$24)$$

Thus we can consider, in evaluation of the logic, only in its standard form of which the highest level of variables in the prefix and the matrix is the same. This means that, in evaluation, the inference algorithm of the many sorted logic, given in Appendix 1, applies at least to the highest level variable. Suppose B and D are given as in the appendix but in the form of ($18) for some variable. Then inference machine apply the conditions of table A1 to the variable at the highest level in testing for $\hat{B} \Rightarrow \hat{D}_i$ and then unify them by the rule given in table A2. To unify variables at the highest level means to make the structure of the objects identical in both formula and, consequently, to make domains of all variables at the lower levels identical ($X_i = Y_i$ in Table A1). Then in testing for $\hat{B} \Rightarrow \hat{D}_i$ in respect to these variables, only the quantifiers are examined for not being the $Q_{Bi} : \exists$ and $Q_{Di} : \forall$. All the remaining procedures are similar to those of many sorted logic.

This is the outline of the inference. We will not include further details in this paper, but we give an additional important theorem.

Theorem 3;

$$(\exists x^n / {*^n} X)(\forall x^{n-1} / x^n) - - - (\forall x^m / x^{m+1})(Q_{m-1} x^{m-1} / x^m) - - -$$
$$- - - [(F\ x^n, \ldots) \cap (G\ x^m, \ldots), \ldots]$$
$$= [(\exists x^m / {*^m} X)(Q_{m-1} x^{m-1} / x^m) - - - (G\ x^m, \ldots)]$$
$$\cap [(\exists x^n / {*^{n-m}} x^m) - - - (F\ x^n, \ldots)]$$

Assume to evaluate ($17). On way of evaluation is to begin at the top of posible structure, that is, to generate $*^3V$ first and then test each element of $*^3V$ for the condition in the main body of ($17). It is, however, almost impossible when the set V is large. Alternatively, we can start at the bottom and go upward. That is, we generate *V, test and leave only those elements of *V that satisfy $(F_L\ z)$. The set of lines, say L, will be generated such as $L \subset *V$ and $|L| << |*V|$ where |L| denotes the number of elements in L. Then, $(F_S\ x)$ is used to obtain S such that $S \subset *L (\subset *^2 V)$ and so on. This might be the way man usually takes in making up a structure. Theorem 3 allows the system to take this procedure.

4.4 Representation and manipulation of data structure

The algorithm for the multi-layer logic is a little more

complicated in the handling of the variables than that of the many sorted logic. However, the reward by the expansion is far exceed the investment, in particular, in its model building capability.

The capability to represent and manipulate the hierarchical structure as well as the graph mentioned before offer us the powerful means for modeling the real objects. Indeed, we can give the meaning to and the theoretical basis for manipulation of the conventional data structure which has been used in many application programs as a software technique but without any solid theoretical basis. Fig.6 illustrates a few examples of structures that can be represented by the technique. These structure will cover: the chemical compounds, the geometrical model, the system assembly and so on.

Once the structure is specified, any kind of description can be given to whole structure, its substructure and components within the framework of multi layer logic. For example, we can define the concept of connection between nodes using (ARC x y) in the example of Fig.6(b) as,

$$(\forall u/X)(\forall v/X)(\forall x/u)(\forall y/v)(\forall m/\text{integer})(\forall n/\text{integer})$$
$$[(\text{ARC } x\ y)\cap(\text{CONNECT } y\ z\ m)\cap(\text{ADD } m\ 1\ n)\Rightarrow(\text{CONNECT } x\ z\ n)] \quad (\$26)$$

where (CONNECT x z n) means that one can go from x to z following the path of length n. For latter convenience, however, we extend it further to the concept of distance. We assume that the relation RMi in Fig.6(b) includes not only the pair of nodes but the distance between them with the definition.

$$(\forall x/X)(\forall z/\text{ident})(\forall u/x)(\forall v/x)(\forall w/\text{real})[(\text{RELATION } z;u\ v\ w)$$
$$\cap(\text{C-FILE2}\ \ x\ z)\Rightarrow(\text{DIST } u\ v\ 1\ w)] \quad (\$27)$$

Then the distance between any two nodes is defined as

$$(\forall u/X)(\forall v/X)(\forall x/u)(\forall y/v)(\forall m/\text{integer})(\forall n/\text{integer})(\forall d/\text{real})(\forall e/\text{real})$$
$$(\forall f/\text{real})[(\text{DIST } x\ y\ 1\ d)\cap(\text{DIST } y\ z\ m\ e)\cap(\text{ADD } m\ 1\ n)\cap(\text{ADD } d\ e\ f)$$
$$\Rightarrow(\text{DIST } x\ z\ n\ f)] \quad (\$28)$$

where (DIST x z n f) means that the length of path and the distance between x and z are n and f respectively.

As another interesting example, consider

$$(\exists u/X)(\forall x/u)(\forall y/u)[(\text{RELATION } \#RQ\ ;\ x\ y)\Rightarrow(\text{ARC } x\ y)]?$$

This is the question asking some u in X (by $(\exists u/X)$) such that , for all elements, x, y of u, if the tuple (x y) is included in the given table, RQ, then there is an arc between x and y. In other words, it is asking which substructure in Fig.6(b) includes a subgraph represented by RQ. The inference machine, then, deduces

$$(\exists u/X)(\exists z/\text{ident})(\forall x/u)(\forall y/u)[\ \tilde{}(\text{RELATION } \#RQ\ ;\ x\ y)$$
$$\cup\{(\text{RELATION } \#RD\ ;\ u\ z)\cap(\text{RELATION } z\ ;\ x\ y)\}] \quad (\$29)$$

This is evaluated as follows;

(1) The relation RQ is read into the working memory.
(2) The relation RD is also read into the working memory. Each item of RD contains the identifier for the relation RZi corresponding to the subset i of X.
(3) The relation RMi is read one by one and is tested for RMi$\supset$RQ according to the rule (iv) of Table 1. If RMi satisfies the condition, then i, the subset name, is an answer.

5. Conclusion

In this paper we have discussed a framework for the problem solving system, which is in many respect different from the conventional information processing style. The center core of the system is the method to build model by means of the extended predicate logic so that both the structure and the attributes of object can be represented in the same framework.

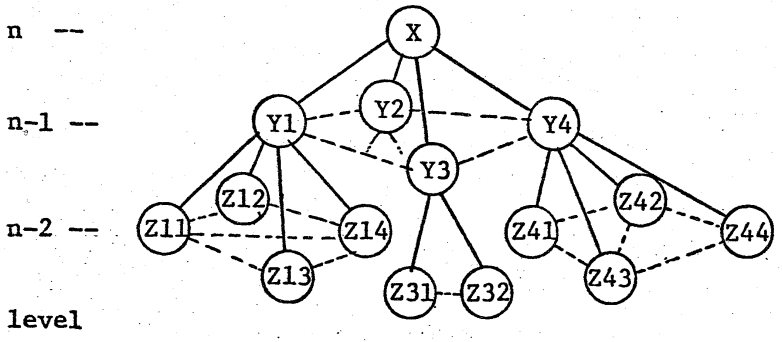
Before end, we give two comments concerning to the efficiency of the system. The system presented in this paper can be used, if one wants, as the expert system which contains expert's knowledge in the applied field[41] and aids non-expert users as a consultant. Our primary object, however, is not in the expert system. Consultation will needs, in general, a deep inference with a lot of information as the knowledge But many problems are remained unsolved, in acquiring and maintaining knowledge, in controling the search and the backtracking, etc.[8][9][10][11][24] Instead, we focussed our attention in use of knowledge to build and manipulate model by providing the system with model description capability. We can expect it being very effective for non-programmer user to solve problem. As an example, assume a user has a transportation problem and need to obtain data concerning to it. The transportation problem may be represented by means of mathematical model, say, a graph, which, in turn is represented by means of database. Today, in using database, he is required himself to have knowledge on what information in what database and how to access it. But if the system has information such as Fig.6(b), he will be permitted to use mathematical terms such as 'arc between x and y' instead of 'the relation name' in expressing his problem. Further if the definition on the distance is given to the system as ($28), he can express his problem in terms of his specific problem. In this case we need only three levels in inference and less problems arises in heuristic operations. In general, at most four or five levels of inferences may produce a large effect in making the computers easy to use. This is the first point we like to stress.

Second, in the system, the problem is automatically transformed and decomposed to the set of built-in functions in the computer. Two kinds of processor are used : inference machine and program execution machine. In order to obtain processing speed, we need to accelerate both processors. Note here that the processing algorithm of both machines are, in principle, independent of specific problems. The inference machine is defined on the basis of the uniform formalism of multi layer logic and the program execution machine is the collection of processing units, each of which is defined to execute a built-in function. Then, most parts of these machines can be translated into hardwares which can work concurrently. In particular, we consider that the use of the database machine is quite desirable.

Many details have been remained undescribed. Of particular importance among them concerning to the subject described in this paper are; organization of knowledge structure, capability to define local world (frame) in the knowledge structure, capability of an instance structure to inherit the attributes of generic description of the structure and so on. We will, however, defer describing these issues to the other paper.

Appendix

Let a formula K in the knowledge base and the query H be given in the forms of $A_1 \cap \ldots \cap A_m \Rightarrow B$ and $D_1 *_1 \ldots *_{n-1} D_n$ respectively where $*_i$, $i=1, 2, \ldots, n-1$, denote i-th connective (either $\cap$ or $\cup$). If necessary, parentheses are to be used to precisely specify the logical structure of H! Suppose $B \Rightarrow D_i$ is proved to hold for some D_i in H. Then, we obtain a new hypothesis H' by replacing D_i in H by $A_1 \cap \ldots \cap A_m$ of K and by modifying the domains and quantifiers of all variables according to a rule described later. That is,



RN

Y1	Y2
Y2	Y4
Y3	Y4
Y1	Y3

RM1

Z11	Z12
Z11	Z14
Z12	Z14
Z13	Z11
Z14	Z13

RY

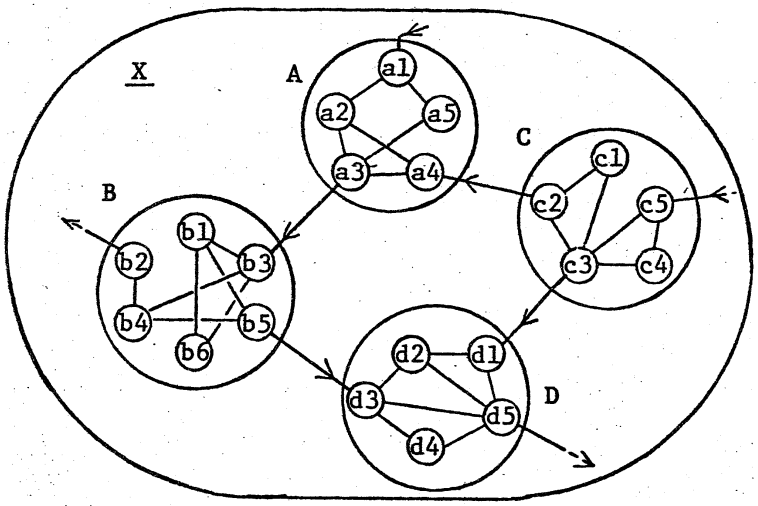
Y1	RM1
Y2	RM2
Y3	RM3
Y4	RM4

RM4

Z41	Z43
Z42	Z41
Z43	Z42
Z43	Z44
Z44	Z42

$(\forall u/X)(\forall v/X)[(\text{RELATION } \#RN\text{: } u\ v) \Rightarrow (\text{ARC } u\ v)]$
$(\forall x/X)(\forall z/id)[(\text{RELATION } \#RY\text{: } x\ z) \Rightarrow (\text{C-FILE } x\ z)]$
$(\forall x/X)(\forall z/id)(\forall u/x)(\forall v/x)[(\text{RELATION } z\text{: } u\ v) \cap (\text{C-FILE } x\ z) \Rightarrow (\text{ARC } u\ v)]$

(a) An example of hierarchical structure



RTA

a1	i
a2	n
a3	o
a4	i
a5	n

RMA

a1	a2
a2	a4
a3	a2
a3	a5
a4	a3
a5	a1

RTB

b1	n
b2	o
b3	i
b4	n
b5	o
b6	n

RMB

b1	b3
b1	b6
b2	b4
b3	b4
b4	b5
b5	b1
b6	b3

RC

A	RTA
B	RTB

RD

A	RMA
B	RMB

RN

A	B
B	D
C	A
C	D

$(\forall x/X)(\forall y/Y)(\forall u/x)(\forall v/y)[(\text{LINK } x\ y) \cap (\text{TYPE } u\ \#o) \cap (\text{TYPE } v\ \#i) \Rightarrow (\text{ARC } u\ v)]$
$(\forall x/X)(\forall y/X)[(\text{RELATION } \#RN\text{: } x\ y) \Rightarrow (\text{LINK } x\ y)]$
$(\forall x/X)(\forall z/id)(\forall u/x)(\forall v/chrct)[(\text{RELATION } z\text{: } u\ v) \cap (\text{C-FILE1 } x\ z) \Rightarrow (\text{TYPE } u\ v)]$
$(\forall x/X)(\forall z/id)[(\text{RELATION } \#RC\text{: } x\ z) \Rightarrow (\text{C-FILE1 } x\ z)]$
$(\forall x/X)(\forall z/id)(\forall u/x)(\forall v/x)[(\text{RELATION } z\text{: } u\ v) \cap (\text{C-FILE2 } x\ z) \Rightarrow (\text{ARC } u\ v)]$
$(\forall x/X)(\forall z/id)[(\text{RELATION } \#RD\text{: } x\ z) \Rightarrow (\text{C-FILE2 } x\ z)]$

(b) An example of recursively defined graph

Fig.6 Examples of typical structure

$$H' = D_1' *_{1} \ldots *_{i-2} D_{i-1}' *_{i-1} (A_1' \cap \ldots \cap A_m') *_{i} D_{i+1}' *_{i+1} \ldots *_{n-1} D_n' \qquad (\$A1)$$

where D_j, $j \neq i$ and A_k, $k=1, 2, \ldots, m$, are D_j and A_k respectively in the original formulas but the domains and quantifiers of variables are modified. H' of formula ($A1) is, in fact, represented by an AND-OR tree which is obtained by replacing the node D_i in the tree of H by the subtree $A_1 \cap \ldots \cap A_m$ of K.

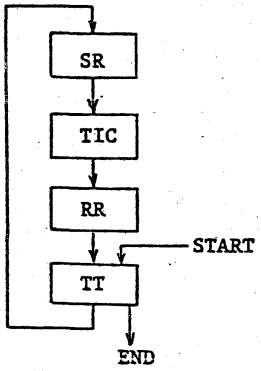


Fig.A1 Deductive inference procedure

Table A1 Implicative condition on individual variable.

Q_{pi}	$Q_{qi}(\bar{Q}_{qi})$	CONDITION
∀	∀(∃)	$X_i \supset Y_i$
∀	∃(∀)	$X_i \wedge Y_i \neq \phi$
∃	∀(∃)	non-implicative
∃	∃(∀)	$X_i \subset Y_i$

Q_{pi}, X_i: Quantifier and domain of i-th variable in P
Q_{qi}, Y_i; Quantifier and domain of i-th variable in Q

If D_i is implied by the simple formula C, i.e., $C \Rightarrow D_i$, then D_i is labeled as T(true) while, if $C \Rightarrow \neg D_i$ is proved, then D_i is labeled as F(false). If none of the cases described above apply for all formulas in the knowledge base, then D is labeled as U(unknown). This is the outline of the deductive procedure. As shown in Fig.A1, it is composed of four blocks named SR, TIC, RR and TT.

SR selects an atom D_i out of H, which includes one or more universally quantified variable(s) (including constants).

TIC, receiving D_i in H from SR, looks for a formula K in the knowledge base that includes a predicate B such that $\hat{B} \Rightarrow \hat{D}_i$ to D_i, where $\hat{B}$ and $\hat{D}_i$ are the single atom formulas obtained from K and H. That is, $\hat{B}$(and $\hat{D}_i$) is obtained by deleting all predicates and variables from K(and H) except those concerning B(and D_i). K is said to satisfy the implicative condition to H.

Let

$$\hat{B} = (Q_{pi_1} x_{i_1} / X_{i_1}) \ldots (Q_{pi_n} x_{i_n} / X_{i_n}) (F\ x_1 \ldots x_n)$$
$$\hat{D}_i = (Q_{q_{j_1}} y_{j_1} / Y_{j_1}) \ldots (Q_{q_{j_n}} y_{j_n} / Y_{j_n}) (F\ y_1 \ldots y_n)$$

Then, $\hat{B} \Rightarrow \hat{D}_i$ if and only if the following two conditions are satisfied (without proof):

(1) For every corresponding variable pair, the relations shown in Table A1 hold between their domains depending on the combinations of quantifiers.

(2) There is no indices pair (i j) such that, in the prefix of $\hat{B}$, they appear in the order of $\ldots(\forall x_i / X_i)\ldots(\exists x_j / X_j)\ldots$ while, in that of $\hat{D}_i$, they appear in the order of $\ldots(\exists y_j / Y_j)\ldots(\forall y_i / Y_i)\ldots$.

Table A2 Derivation rule for quantifier and domain of new query.

	Q_{Ki}	$Q_{Hi}(\bar{Q}_{Hi})$	$Q'_{Hi}(\bar{Q}'_{Hi})$	Z_i
$x_i/y_i \in x_0$	∀	∀(∃)	∀(∃)	Y_i
	∀	∃(∀)	∃(∀)	$X_i \wedge Y_i$
	∃	∃(∀)	∀(∃)	X_i
$x_i/y_i \in x_1$	—	∀(∃)	∀(∃)	Y_i
	—	∃(∀)	∃(∀)	Y_i
	∃	—	∀(∃)	X_i
	∀	—	∃(∀)	X_i

Q_{Ki}, X_i; Quantifier and domain of variable in K corresponding to i-th variable in H'
Q_{Hi}, Y_i; Quantifier and domain of variable in H corresponding to i-th variable in H'
Q'_{Hi}, Z_i; Quantifier and domain of i-th variable in H'

In practice, ~H, the negation of H, is used instead of H for deductive operations as the processing is simpler. In the text however, the process is described using a positive H for ease of explanation.

The set of all variables included in either or both H and K may be divided into two classes: those included in D_i with their

corresponding variables in K and the remaining variables. These sets are denoted x_0 and x_1 respectively. The domain and quantifier of each variable in the new hypothesis H' of the form of formula ($A|1) is obtained for each combination of the quantifiers of the corresponding variables in K and H as shown in Table A2. In the deductive process, RR takes charge of generating the new hypothesis.

TT watches whether or not the process terminates. The deductive procedure terminates either when the logical value of the query can be determined or when the formula is resulted, which is composed only of the procedural predicates described in Sec.3.3.

REFERENCES

1 Berztiss, A.T., Data Structures, in: Theory and Practice (Academic Press, 1971).

2 Bobrow, D.G. and Winograd, T., An Overview of KRL a Knowledge Representation Language, Cognitive Science 1-1 (Jan. 1977).

3 Bobrow, D.G., Winograd, T. et.al., Experience with KRL-0-One cycle of a Knowledge Representation Language, Pro. 5th Int'l. Joint Conf. on Airtificial Intelligence (1977).

4 Chang, C.L. and Lee, R.C.T., Symbolic Logic and Mechanical Theorem Proving (Academic Press, 1973).

5 Codd, E.F., A Relational Model of Data for Large Shared Data Banks, Comm. ACM 13-6 (1970) 377-397.

6 Codd, E.F., A Data Base Sublanguage Found on the Relational Calculus, Proc. ACM SIGFIDET Workshop on Data Description and Control (1971).

7 Date, C.J., An Introduction to Data Base Systems (Adison-Wesley, 1976).

8 Davis, M., Non-Monotonic Logic I, Artificial Intelligence 13-1,2 (April 1980) 27-40.

9 Davis, R., Meta-Rules: Reasoning about Control, Artificial Intelligence 15-3 (Dec.1980) 179-222.

10 Davis, R., Content Reference: Reasoning about Rules, Artificial Intelligence 15-3 (Dec.1980) 223-240.

11 Doyle, J., A Grimpse of Truth Maintenance, Proc. 6th Int'l Joint Conf. on Artificial Intelligence 232-237.

12 Fitter, M., Towards More Natural Interactive Systems, Int'l.J. Man-Machine Studies 11 (1979) 339-350.

13 Gains, B.R., Logical Foundations for Database Systems, Int'l.J. Man-Machine Studies 11 (1979) 481-500.

14 Green, C., The Use of Theorem Proving Techniques in Question Answering System, Proc. 23rd National Conf. ACM, (Brandon Press, 1968).

15 Harrison, M.C., Data-Structures and Programming (Scott, Foresman and Co. 1973).

16 Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D. and Slocum, J., Developing a Natural Language Interface to Complex Data, ACM Transaction on Database Systems 3-2 (June 1978).

17 Hendrix, G.G., Expanding the Utility of Semantic Networks Through Partitioning, Proc.4th Int'l Joint Conf. on Artificial Intelligence (1975).

18 Knuth, D.E., The Art of Programming, Vol.1, Fundamental Algorithm, Chap.2, (Addison Wesley, 1973) 228-616.

19 Kowalski, R.A., Predicate Logic as Programming Language, Proc. IFIP 74 (North Holland, 1974) 569-574.

20 Latombe, J.C., Artificial Intelligence in Computer-Aided Design:

The "TROPIC" System, in: Allan, J.J. (eds.), CAD System (North-Holland, 1977) 61-120.
21 Ledgard, H., Whiteside, J.A., Singer, A. and Seymour, W., The Natural Language of Interactive Systems, CACM 23-10 (Oct. 1980) 556-563.
22 Lewis, W.P., The Role of Intelligence in the Design of Mechanical Computers, Proc. Man-Machine Communication in CAD/CAM IFIP W.G. 5.2-W.G.5.3 Working Conference (Oct. 1980) 47-72.
23 Matsuka, H. and Uno, S., Application of Advanced Integrated Designes Activity Support System, Proc. Man-Machine Communication in CAD/CAM IFIP W.G.5.2-W.G.5.3 Working Conference (Oct. 1980) 176-191.
24 McDermott, D. and Doyle, J., Circumscription- A Form of non-Monotonic Reasoning, Artificial Intelligence 13-1 & 2, (April 1980) 27-40.
25 Mendelson, E., Introduction to Mathematical Logic (Van Nostrand Reinhold, 1964).
26 Minker, J., An Experimental Relational Data Base System Based on Logic, in: Gallaire, H. and Minker, J. (eds.), Logic and Database (Prenum Press, 1978) 107-148.
27 Myloponles, J., Cohen, P., Borgida, A. and Suger, L., Semantic Networks and the Generation of Context, Proc. 4th Int'l. Joint Conf. on Artificial Intelligence (1975).
28 Newell, A., Production System : Models of Control Structures, in: Chase, W.C.(ed), Visual Information Processing (Academic Press, 1973) 463-526.
29 Newell, R.G., Sancha, T.L., Williamson, R.M. and Hiles, J.O., The Design of System for CAD, in: Allan, J.J. (ed.), CAD System (North-Holland, 1977) 121-142.
30 Newell, M. and Evans, D.C., Modeling by Computer, in: Allan, J.J. (ed.), CAD System (North-Holland, 1977) 291-308.
31 Newman, W.M. and Sproull, R.F., Principles of Interactive Computer Graphics (Second ed.), (McGraw-Hill).
32 Ohsuga, S., Toward Intelligent Interactove Systems, in: Guedj, R. A. et al. (eds.), Methodology of interaction (1978) 339-360.
33 Ohsuga, S., Perspectives on New Computer Systems of the Next Generation - A Proposal for Knowledge-Based Systems, Jrnl. of Information Processing 3-3 (Sept. 1980) 171-185.
34 Reiter,R., Deductive Question-Answering on Relational Data Bases, in: Gallaire, H. and Minker, J. (eds.), Logic and Database (Prenum Press, 1978) 149-178.
35 Reiter, R., The Mathematics of Non-Monotonic Reasoning, Artificial Intelligence 13-1,2 (April, 1980) 73-80.
36 Robinson, J.A., A Machine-Oriented Logic Based on the Resolution Principle, JACM 12 (Jan. 1965).
37 Sacerdoti, E.D., Language Access to Distributed Data with Error Recovery, Proc.5th Int'l. Joint Conf. on Artificial Intelligence, (August 1977).
38 Sagalowicz, IDA: An Intelligent Data Access Program, Proc. Third Int'l. Conf. on Very Large Data Bases (Oct. 1977).
39 Sambura, A.S., Conceptual Framework for Man-Machine Communication, Proc. Man-Machine Communication in CAD/CAM IFIP W.G.5.2-W.G.5.3 Working Conference (Oct. 1980) 19-27.
40 Schank, R. and RiegerII, C.J., Inference and the Computer Understanding of Natural Language, Artificial Intelligence 5 (1974) 373-412.
41 Shortliffe, E.H., Computer-Based Medical Consultations : MYCIN (American Elesvier, 1976).
42 Trembley, J.P. and Sorenson, P.G., An Introduction to Data Structures with Applications (McGraw Hill, 1976).

43 Valle, G., Relational Data Handling Techniques in Computer Aided Design Procedures, in: Allan, J.J. (ed.), CAD System (North-Holland, 1977) 309-326.
44 Warman, E.A., Man in a Machine Environment, Proc. Man-Machine Communication in CAD/CAM IFIP W.G.5.2-W.G.5.3 Working Conference (Oct. 1980) 1-13.